

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号
特開2000-242614
(P2000-242614A)

(43)公開日 平成12年9月8日(2000.9.8)

(51)Int.Cl.⁷

G 0 6 F 15/177

識別記号

6 7 4

F I

G 0 6 F 15/177

テーマコード(参考)

6 7 4 A 5 B 0 4 5

審査請求 未請求 請求項の数28 O L (全 13 頁)

(21)出願番号

特願平11-43713

(22)出願日

平成11年2月22日(1999.2.22)

(71)出願人 000006655

新日本製鐵株式会社

東京都千代田区大手町2丁目6番3号

(72)発明者 永島 聡

東京都千代田区大手町2-6-3 新日本
製鐵株式会社内

(72)発明者 濱田 和郎

東京都千代田区大手町2-6-3 新日本
製鐵株式会社内

(74)代理人 100090273

弁理士 國分 孝悦

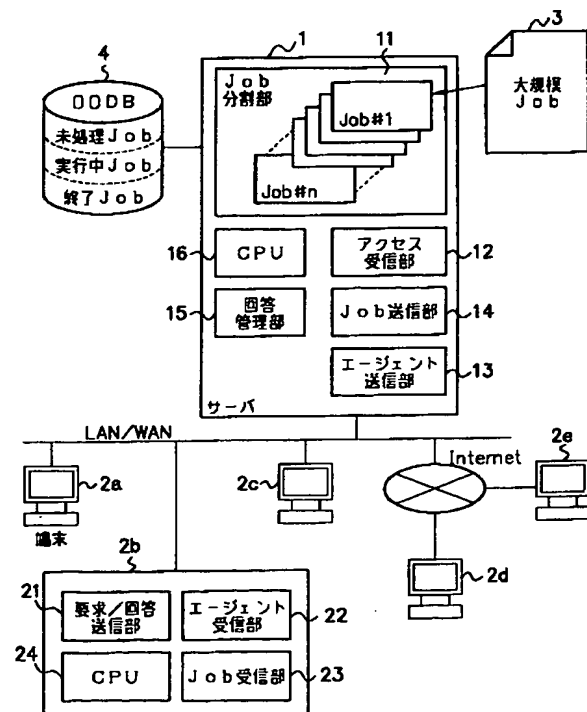
Fターム(参考) 5B045 BB01 BB19 BB47 GG02

(54)【発明の名称】 分散処理システムおよびその方法、分散処理を行うための端末装置および記録媒体

(57)【要約】

【課題】 ネットワーク上で密結合されたCPU群を有効に利用して大規模な処理を少ない負荷で行うことができるようにする。

【解決手段】 大規模ジョブ3を複数の小ジョブ#1～#nに分割するジョブ分割部11と、分割された複数の小ジョブ#1～#nのうち未処理のジョブを、各端末2a～2eのうちジョブ取得要求を行った端末に対して与えるジョブ送信部14とを備え、大規模ジョブ3を複数の小ジョブ#1～#nに分割してそれぞれを複数の端末2a～2eに配信して分散実行させ、それぞれの回答を各端末2a～2eからサーバ1に返すようにすることにより、大規模ジョブ3の処理を少ない計算負荷で実行することができるようにする。また、端末2a～2eから要求があったときのみ分割ジョブの配信を実行することにより、各計算機のCPUパワーを常に開放しておくなくても済むようにする。



BEST AVAILABLE COPY

【特許請求の範囲】

【請求項 1】 ネットワーク上に分散して存在する複数の計算機を利用してジョブを分散実行する分散処理システムであって、

上記複数の計算機のうちジョブ取得要求が行われた計算機または利用可能な状態にある計算機に対して、上記ジョブから分割された複数の分割ジョブのうち未処理の分割ジョブを与えるジョブ配信手段を備えることを特徴とする分散処理システム。

【請求項 2】 上記ジョブ配信手段は、上記ジョブ取得要求が行われた計算機または利用可能な状態にある計算機に対して、当該計算機に常駐して所定の処理を行うエージェントを与えるエージェント送信手段と、上記計算機に常駐したエージェントからの要求に応じて、上記未処理の分割ジョブを与えるようにする分割ジョブ送信手段とを備えることを特徴とする請求項 1 に記載の分散処理システム。

【請求項 3】 上記計算機に常駐したエージェントは、与えられた分割ジョブの処理が終わる毎に次のジョブ取得要求を行うようになされていることを特徴とする請求項 2 に記載の分散処理システム。

【請求項 4】 上記計算機の状態に応じて、上記未処理の分割ジョブの配信を制御するようにしたことを特徴とする請求項 1～3 の何れか 1 項に記載の分散処理システム。

【請求項 5】 上記計算機に分割ジョブを送信してからその処理結果である回答が返ってくるまでのタイムアウト時間を管理し、上記タイムアウト時間内に回答が返ってこなかった分割ジョブについては、もう一度配信し直すようにする回答管理手段を備えることを特徴とする請求項 4 に記載の分散処理システム。

【請求項 6】 上記タイムアウト時間は、個々の分割ジョブに応じて異なるように設定されることを特徴とする請求項 5 に記載の分散処理システム。

【請求項 7】 上記計算機に分割ジョブを送信してからその処理結果である回答が返ってくるまでの応答時間に依拠して、上記計算機に与えるジョブ数を可変とする送信ジョブ数管理手段を備えることを特徴とする請求項 4 に記載の分散処理システム。

【請求項 8】 上記分割ジョブの配信を上記計算機からの要求に応じて受動的に行う方式、上記分割ジョブの配信を上記利用可能な状態にある計算機に対して能動的に行う方式の上記計算機上での採用状態に応じて、上記分割ジョブの配信に制約を与えるジョブ配信制約手段を備えることを特徴とする請求項 4 に記載の分散処理システム。

【請求項 9】 上記計算機は、与えられた分割ジョブの実行をバックグラウンド処理にて行うことを特徴とする請求項 1～8 の何れか 1 項に記載の分散処理システム。

【請求項 10】 上記計算機は、与えられた分割ジョブ

の実行中であることをユーザに知らせるための報知手段を備えることを特徴とする請求項 1～8 の何れか 1 項に記載の分散処理システム。

【請求項 11】 上記分散処理システムはオブジェクト指向アーキテクチャに基づき構築されることを特徴とする請求項 1～10 の何れか 1 項に記載の分散処理システム。

【請求項 12】 上記ジョブ配信手段は、上記ネットワーク上に分散して存在する複数の計算機のうちの 1 つまたは専用のジョブ管理端末装置内に備えられることを特徴とする請求項 1～11 の何れか 1 項に記載の分散処理システム。

【請求項 13】 ネットワーク上に分散して存在する複数の計算機を利用してジョブを分散実行する分散処理方法であって、

上記複数の計算機のうちジョブ取得要求が行われた計算機または利用可能な状態にある計算機に対して、上記ジョブから分割された複数の分割ジョブのうち未処理の分割ジョブを与え、

上記計算機に与えられた分割ジョブをその計算機上で処理し、その処理結果である回答を上記分割ジョブの供給元に返すようにしたことを特徴とする分散処理方法。

【請求項 14】 上記分割ジョブの供給元では、上記ジョブ取得要求が行われた計算機または利用可能な状態にある計算機に対して、当該計算機に常駐して所定の処理を行うエージェントを与え、上記計算機上に常駐したエージェントからの要求に応じて、上記未処理の分割ジョブを与えるようにしたことを特徴とする請求項 13 に記載の分散処理方法。

【請求項 15】 上記分割ジョブの供給元では、上記計算機の状態に応じて、上記未処理の分割ジョブの配信を制御するようにしたことを特徴とする請求項 13 または 14 に記載の分散処理方法。

【請求項 16】 上記計算機に分割ジョブを送信してからその回答が返ってくるまでのタイムアウト時間を管理し、上記タイムアウト時間内に回答が返ってこなかった分割ジョブについては、もう一度配信し直すようにすることを特徴とする請求項 15 に記載の分散処理方法。

【請求項 17】 上記計算機に分割ジョブを送信してからその回答が返ってくるまでの応答時間に依拠して、上記計算機に与えるジョブ数を可変とするようにしたことを特徴とする請求項 15 に記載の分散処理方法。

【請求項 18】 上記分割ジョブの配信を上記計算機からの要求に応じて受動的に行う方式、上記分割ジョブの配信を上記利用可能な計算機に対して能動的に行う方式の上記計算機上での採用状態に応じて、上記分割ジョブの配信に制約を与えるようにしたことを特徴とする請求項 15 に記載の分散処理方法。

【請求項 19】 ネットワーク上に分散して存在する複数の計算機を利用してジョブの分散処理を行うための端

末装置であって、上記複数の計算機のうちジョブ取得要求が行われた計算機または利用可能な状態にある計算機に対して、上記ジョブから分割された複数の分割ジョブのうち未処理の分割ジョブを与えるジョブ配信手段を備えることを特徴とする分散処理を行うための端末装置。

【請求項 20】 上記計算機の状態に応じて、上記未処理の分割ジョブの配信を制御するようにしたことを特徴とする請求項 19 に記載の分散処理を行うための端末装置。

【請求項 21】 上記計算機に分割ジョブを送信してからその処理結果である回答が返ってくるまでのタイムアウト時間を管理し、上記タイムアウト時間内に回答が返ってこなかった分割ジョブについては、もう一度配信し直すようにする回答管理手段を備えることを特徴とする請求項 20 に記載の分散処理を行うための端末装置。

【請求項 22】 上記計算機に分割ジョブを送信してからその処理結果である回答が返ってくるまでの応答時間に応じて、上記計算機に与えるジョブ数を可変とする送信ジョブ数管理手段を備えることを特徴とする請求項 20 に記載の分散処理を行うための端末装置。

【請求項 23】 上記分割ジョブの配信を上記計算機からの要求に応じて受動的に行う方式、上記分割ジョブの配信を上記利用可能な状態にある計算機に対して能動的に行う方式の上記計算機上での採用状態に応じて、上記分割ジョブの配信に制約を与えるジョブ配信制約手段を備えることを特徴とする請求項 20 に記載の分散処理を行うための端末装置。

【請求項 24】 ネットワーク上に分散して存在する複数の計算機を利用してジョブを分散実行する分散処理システムにおいて、
上記複数の計算機のうちジョブ取得要求が行われた計算機または利用可能な状態にある計算機に対して、上記ジョブから分割された複数の分割ジョブのうち未処理の分割ジョブを与えるジョブ配信手段としてコンピュータを機能させるためのプログラムを記録したことを特徴とするコンピュータ読み取り可能な記録媒体。

【請求項 25】 上記計算機の状態に応じて、上記未処理の分割ジョブの配信を制御するようにするための機能を上記コンピュータに実現させるためのプログラムを更に記録したことを特徴とする請求項 24 に記載のコンピュータ読み取り可能な記録媒体。

【請求項 26】 上記計算機に分割ジョブを送信してからその処理結果である回答が返ってくるまでのタイムアウト時間を管理し、上記タイムアウト時間内に回答が返ってこなかった分割ジョブについては、もう一度配信し直すようにする回答管理手段としての機能を上記コンピュータに実現させるためのプログラムを記録したことを特徴とする請求項 25 に記載のコンピュータ読み取り可能な記録媒体。

【請求項 27】 上記計算機に分割ジョブを送信してか

らその処理結果である回答が返ってくるまでの応答時間に応じて、上記計算機に与えるジョブ数を可変とする送信ジョブ数管理手段としての機能を上記コンピュータに実現させるためのプログラムを記録したことを特徴とする請求項 25 に記載のコンピュータ読み取り可能な記録媒体。

【請求項 28】 上記分割ジョブの配信を上記計算機からの要求に応じて受動的に行う方式、上記分割ジョブの配信を上記利用可能な状態にある計算機に対して能動的に行う方式の上記計算機上での採用状態に応じて、上記分割ジョブの配信に制約を与えるジョブ配信制約手段としての機能を上記コンピュータに実現させるためのプログラムを記録したことを特徴とする請求項 25 に記載のコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は分散処理システムおよびその方法、分散処理を行うための端末装置および記録媒体に関し、特に、ネットワーク上に分散して存在する複数の計算機パワーを利用して大規模な処理を分散して行うためのシステムに用いて好適なものである。

【0002】

【従来の技術】 近年、コンピュータで行う処理の高度化および複雑化などに伴い、その処理プログラムが膨大なものとなってきている。個々の計算機の性能は、以前に比べて格段に向上しているが、それでもこのような大規模な処理を 1 台の計算機で実行するには、その計算負荷が極めて大きなものとなっている。従来、大規模な計算問題を扱うために、処理能力が極めて高いスーパーコンピュータ等を使用することもあるが、これは非常に高価であり、一般的に使えるものではなかった。

【0003】

【発明が解決しようとする課題】 一方、近年では、LAN (Local Area Network) や WAN (Wide Area Network)、あるいはインターネット等のネットワークを用いることにより、複数の計算機の結びつきは益々密となり、また、相互接続の範囲も広がってきている。ネットワーク上にあるこれら複数の計算機を相互接続された大 CPU 群と見た場合、その処理能力は極めて高いものと言える。

【0004】 しかしながら、現状では、このように相互接続された処理能力の高い計算機資源を十分に生かし切れていないのが実状である。例えば、インターネットに関して言えば、ネットワーク上の大部分を占める一般のユーザが享受しているサービスは、Web によるホームページサーフィンや e-mail 程度である。また、ネットワーク上に接続されているが、何の処理も行われずに遊んでいる空 CPU が多いのも事実である。

【0005】 そのため、1 台の計算機で処理をさせるには負荷が大き過ぎると考えられる大規模な計算処理を複

数の小さなジョブに分割し、これらをネットワーク上の複数の計算機が備えるCPU群を用いて分散して行うようにすれば、ネットワーク上の余った計算機資源を有効に活用することが可能となる。そこで従来、インターネット上に分散して存在する複数の計算機を協調して動作させることにより、大規模な並列計算を行わせようとするシステムが提案されてきている。

【0006】ところが、上記従来の分散処理システムは、ネットワーク接続されている複数の計算機を強制的に協調動作させるものであった。そのため、ネットワーク上の各計算機は、他の計算機で定義された計算問題等を解くためのCPUパワーを常時提供しなければならず、自己の計算機内で通常の処理を行う際に意図せず余計な負荷がかかってしまうことがあるという問題があった。

【0007】本発明は、このような実情に鑑みて成されたものであり、1台の計算機で処理するには負荷が大き過ぎる大規模な処理を、ネットワーク上で密結合された各計算機のCPU群を有効に利用して少ない負荷で実行できるようにすることを目的とする。また、本発明は、ネットワーク上に接続された計算機自体に余計な負担をかけないようにすることを目的とする。

【0008】

【課題を解決するための手段】本発明の分散処理システムは、ネットワーク上に分散して存在する複数の計算機を利用してジョブを分散実行する分散処理システムであって、上記複数の計算機のうちジョブ取得要求が行われた計算機または利用可能な状態にある計算機に対して、上記ジョブから分割された複数の分割ジョブのうち未処理の分割ジョブを与えるジョブ配信手段を備えることを特徴とする。

【0009】ここで、上記ジョブ配信手段は、例えば、上記ジョブ取得要求が行われた計算機または利用可能な状態にある計算機に対して、当該計算機に常駐して所定の処理を行うエージェントを与えるエージェント送信手段と、上記計算機に常駐したエージェントからの要求に応じて、上記未処理の分割ジョブを与えるようにする分割ジョブ送信手段とを備えて構成される。上記計算機に常駐したエージェントは、与えられた分割ジョブの処理が終わる毎に次のジョブ取得要求を行うようになされている。

【0010】本発明の他の態様では、上記計算機の状態に応じて、上記未処理の分割ジョブの配信を制御するようにしたことを特徴とする。例えば、上記計算機に分割ジョブを送信してからその処理結果である回答が返ってくるまでのタイムアウト時間を管理し、上記タイムアウト時間内に回答が返ってこなかった分割ジョブについては、もう一度配信し直すようにする回答管理手段を備える。ここで、上記タイムアウト時間は、個々の分割ジョブに応じて異なるように設定されるようにしても良い。

また、上記計算機に分割ジョブを送信してからその処理結果である回答が返ってくるまでの応答時間に応じて、上記計算機に与えるジョブ数を可変とする送信ジョブ数管理手段を備えても良い。また、上記分割ジョブの配信を上記計算機からの要求に応じて受動的に行う方式、上記分割ジョブの配信を上記利用可能な状態にある計算機に対して能動的に行う方式の上記計算機上での採用状態に応じて、上記分割ジョブの配信に制約を与えるジョブ配信制約手段を備えても良い。

10 【0011】また、上記計算機は、与えられた分割ジョブの実行をバックグラウンド処理にて行うようにしても良いし、与えられた分割ジョブの実行中であることをユーザに知らせるようにしても良い。また、上記分散処理システムはオブジェクト指向アーキテクチャに基づき構築されるようにしても良い。さらに、上記ジョブ配信手段は、上記ネットワーク上に分散して存在する複数の計算機のうちの1つまたは専用のジョブ管理端末装置内に備えられるようにしても良い。

20 【0012】また、本発明の分散処理方法は、ネットワーク上に分散して存在する複数の計算機を利用してジョブを分散実行する分散処理方法であって、上記複数の計算機のうちジョブ取得要求が行われた計算機または利用可能な状態にある計算機に対して、上記ジョブから分割された複数の分割ジョブのうち未処理の分割ジョブを与え、上記計算機に与えられた分割ジョブをその計算機上で処理し、その処理結果である回答を上記分割ジョブの供給元に返すようにしたことを特徴とする。本発明の他の態様では、上記分割ジョブの供給元において、上記計算機の状態に応じて、上記未処理の分割ジョブの配信を制御するようにしても良い。

30 【0013】また、本発明の分散処理を行うための端末装置は、ネットワーク上に分散して存在する複数の計算機を利用してジョブの分散処理を行うための端末装置であって、上記複数の計算機のうちジョブ取得要求が行われた計算機または利用可能な状態にある計算機に対して、上記ジョブから分割された複数の分割ジョブのうち未処理の分割ジョブを与えるジョブ配信手段を備えることを特徴とする。本発明の他の態様では、上記計算機の状態に応じて、上記未処理の分割ジョブの配信を制御するようにしても良い。

40 【0014】また、本発明のコンピュータ読み取り可能な記録媒体は、ネットワーク上に分散して存在する複数の計算機を利用してジョブを分散実行する分散処理システムにおいて、上記複数の計算機のうちジョブ取得要求が行われた計算機または利用可能な状態にある計算機に対して、上記ジョブから分割された複数の分割ジョブのうち未処理の分割ジョブを与えるジョブ配信手段としてコンピュータを機能させるためのプログラムを記録したことを特徴とする。本発明の他の態様では、上記計算機の状態に応じて、上記未処理の分割ジョブの配信を制御

するようにするための機能を上記コンピュータに実現させるためのプログラムを更に記録したことを特徴とする。

【0015】上記のように構成した本発明によれば、ネットワーク上に存在する複数の計算機のうち、ジョブ取得要求があった計算機、あるいは利用可能な状態となっている計算機に対して、分割された小さなジョブが送られて実行されることとなるので、各計算機のCPUパワーを外部から与えられるジョブに対して常に開放しておくことなく、大規模なジョブを複数の計算機を利用して分散実行することが可能となる。

【0016】

【発明の実施の形態】（第1の実施形態）図1は、第1の実施形態による分散処理システムの構成を示すブロック図である。図1に示すように、本実施形態の分散処理システムでは、LANやWANあるいはインターネット等のネットワーク上に、複数の計算機が分散して存在している。すなわち、ジョブの管理を行うジョブ管理端末装置であるサーバ1と、ユーザが通常使用する端末2a, 2b, 2c, 2d, 2e, …とがネットワークを介して相互に接続されており、互いにデータの送受信を行うことができるようになっている。なお、図1では端末を5つのみ示しているが、実際にはこれより多くの端末が存在している。

【0017】本実施形態において、上記サーバ1は、分割可能な大規模ジョブ3を適宜分割し、ネットワーク上に存在する複数の端末2a～2eに対して分割済の小さなジョブ#1～#nを割り当てて解いてもらう。そして、それぞれの端末2a～2eから小ジョブ#1～#nの処理結果である回答を集めて、最初の大規模ジョブ3の解答とする。このような動作の実現方法として、第1の実施形態では、Applet方式を採用する。以下に、上記サーバ1および各端末2a～2eの構成を詳しく説明する。

【0018】まず、サーバ1の構成において、ジョブ分割部11は、与えられた大規模ジョブ3を複数の小ジョブ#1～#nに分割する。大規模ジョブ3は、例えばオブジェクト指向プログラムで書かれた計算問題等であり、複数の小ジョブ#1～#nに分割可能なものである。大規模ジョブ3をどのように分割するかは、その大規模ジョブ3のプログラム自体に書かれている（ユーザが記述する）。ジョブ分割部11は、その記述された内容に従って、大規模ジョブ3を複数の小ジョブ#1～#nに分割することになる。

【0019】ここで、分割の仕方としては、例えば、大規模ジョブ3を機能モジュール（オブジェクト）毎に分割してそれぞれを各端末2a～2eに割り当てる方法や、大規模ジョブ3が問題に対するパラメータの与え方を様々に変えることによって最適解を得るような探索問題の場合に、パラメータの与え方を各端末2a～2e毎

に変えることによって処理を分割する方法などが考えられる。

【0020】分割された各小ジョブ#1～#nは、オブジェクト指向データベース（OODB）4内に、未処理ジョブ、実行中ジョブ、実行終了ジョブに整理して格納される（例えば、どの分割ジョブが未処理、実行中あるいは実行済であるかを表したリスト情報によって整理される）。なお、分割された当初は全て未処理ジョブである。

10 【0021】アクセス受信部12は、各端末2a～2eからネットワークを介して与えられる各種のアクセス要求を受信するものである。例えば、サーバ1上で立ち上がっているWebのホームページに対するアクセス要求を受信する。エージェント送信部13は、各端末2a～2eのうち、上記ホームページに対するアクセス要求を行った端末に対して、後述するエージェント（Applet）を与えるものである。また、ジョブ送信部14は、各端末2a～2eのうちジョブ取得の要求が行われた端末に対して未処理の分割ジョブを与えるものである。

20 【0022】また、回答管理部15は、図2に示すような回答管理テーブルを有し、サーバ1から各端末2a～2eに小ジョブ#1～#nを配信してからその処理結果である回答を所定のタイムアウト時間内に受信したかどうかを管理し、上記OODB4内に格納されている各小ジョブ#1～#nの端末2a～2eに対する配信制御を行う。以下、図2の回答管理テーブルについて説明する。

30 【0023】図2において、ジョブIDは、大規模ジョブ3から分割されたそれぞれの小ジョブ#1～#nに対して付されるユニークなシリアルナンバーである。回答管理部15は、ある未処理の小ジョブがジョブ送信部14からある端末に送信された場合、その小ジョブを未処理ジョブのリストから実行中ジョブのリストに移す。また、本実施形態では、サーバ1から端末2a～2eに小ジョブを送信してからその回答が返ってくるまでのタイムアウト時間として、例えば5分を設定しており、そのタイムアウト時間内に回答が返ってきた場合は、その小ジョブを実行中ジョブのリストから実行終了ジョブのリストに移す。

40 【0024】図2中にNo. 1、No. 3で示される小ジョブ（小ジョブ#1～#nの中の何れか2つ）は、タイムアウト時間内に回答が返ってきたので、実行終了ジョブのリストに移されている。一方、No. 2で示される小ジョブは、タイムアウト時間を過ぎても回答が返ってこなかったため、処理が終了しなかったとして、実行中ジョブのリストから未処理ジョブのリストに戻される。

50 【0025】このように未処理ジョブのリストに戻されることにより、その小ジョブはその後再び何れかの端末に送られることになる。この場合、その小ジョブが前回送られた端末上からその小ジョブの処理結果である回答

がタイムアウト時間の経過後に送られてくると、新たに小ジョブが送り直された端末上から返された回答と重複してしまうことがある。

【0026】そこで、実行中ジョブのリストから未処理ジョブのリストに戻す際に、同じ小ジョブに対して前回のNo. 2とは異なる新たなシリアルナンバーを付すとともに、No. 2のシリアルナンバーは今後無視するようにすることにより、前回No. 2のシリアルナンバーが付されて送信された小ジョブの回答が長い時間経ってから返ってきたとしても、その回答は無視して、新たなシリアルナンバーを付して送り直した方の回答を採用するようにすることができる。

【0027】回答管理部15は、このような回答管理テーブルを随時更新しながら参照することにより、今どの小ジョブが未処理として残っているかを把握して、ジョブ送信部14から各端末2a～2eに送信する小ジョブを管理する。なお、ここでは、各小ジョブ#1～#nに対するタイムアウト時間を一律に5分に設定しているが、小ジョブ毎に処理負荷の大きさが異なることも考えられる。このような場合に対応するために、個々の小ジョブ毎にタイムアウト時間を個別に設定するようにしても良い。

【0028】以上に説明したサーバ1内のジョブ分割部11、アクセス受信部12、エージェント送信部13、ジョブ送信部14および回答管理部15の処理は、実際にはCPU16によって実行される。

【0029】次に、各端末2a～2eを代表して、端末2bの構成を説明する。要求/回答送信部21は、ネットワークを介してサーバ1に対して各種の要求や小ジョブ実行の結果である回答を送信するものである。例えば、サーバ1上で立ち上がっているWebのホームページに対するアクセス要求を送信したり、端末2b上に常駐したエージェントから小ジョブ#1～#nの取得要求を送信したりする。また、受信した小ジョブを実行した結果得られる回答も送信する。

【0030】エージェント受信部22は、上記ホームページに対するアクセス要求を行った結果としてサーバ1から送られてくるエージェントを受信するものである。また、ジョブ受信部23は、端末2b上に常駐したエージェントからのジョブ取得要求に応じて、サーバ1から送られてくる未処理の小ジョブを受信するものである。CPU24は、上記要求/回答送信部21、エージェント受信部22およびジョブ受信部23の処理を実行するとともに、常駐したエージェントに規定されているプログラム処理や、受信した小ジョブに規定されているプログラム処理を実行する。

【0031】本実施形態のエージェントは、サーバ1からダウンロードされた先の端末2a～2eに常駐し、そのCPUパワーを利用して所定の処理を行うプログラムであり、例えばJAVAScriptのAppletで構成される。

JAVAScriptのAppletは、計算機のOS等に依存することなく動作可能であるだけでなく、規定された処理以外は実行しない、言い換えれば、基本的には常駐先のデータベースやファイル中に忍び込んでデータを盗んだり改竄することができず、安全性が高いので、好ましい。

【0032】ここで、本実施形態のエージェントに規定されている機能について説明する。本実施形態のエージェントは、少なくとも小ジョブ#1～#nの取得と回答の返信の機能を有している。すなわち、端末2a～2e上に常駐したエージェントがサーバ1側に小ジョブ#1～#nの送信を要求すると、未処理の小ジョブがサーバ1から送られてくる。エージェントは、この送られてきた小ジョブの処理をCPU24に実行させ、その回答をサーバ1に返す。エージェントは、1つの小ジョブの処理が終了すると、サーバ1側に小ジョブ#1～#nの送信を再び要求し、次の小ジョブの処理へと進む。

【0033】次に、以上のように構成した本実施形態による分散処理システム中のサーバ1が行うフレームワーク（ユーザが大規模ジョブ3の分割方法をプログラム上に記述すること以外のサーバ1が行う動作）を、図3および図4を参照しながら説明する。なお、図3は、サーバ1の全体の動作の流れを示すフローチャートであり、図4は、図3中のステップS5における動作の中身を示すフローチャートである。

【0034】図3において、まず最初に、サーバ1は所定の管理情報をセットする（ステップS1）。ここでは、例えば図2に示したような回答管理テーブルを初期状態にセットする。次に、本実施形態のフレームワークにひも付いたWebのホームページがサーバ1上で立ち上がっているかどうかを判断し（ステップS2）、立ち上がっていない場合は立ち上げる（ステップS3）。そして、サーバ1内のジョブ分割部11が、大規模ジョブ3に記述された分割方法に従って、当該大規模ジョブ3を複数の小ジョブ#1～#nに分割する（ステップS4）。

【0035】以上のようにして、大規模ジョブ3が複数の小ジョブ#1～#nに分割されてOODB4に格納されることによって事前準備が完了すると、稼働しているフレームワークのプログラムによってジョブ配信が実行される（ステップS5）。すなわち、図4に示すように、まず、各端末2a～2eの要求/回答送信部21からサーバ1のアクセス受信部12に対して、ホームページへのアクセスがあったかどうかを判断する（ステップS11）。このアクセスがない場合は、所定のディレイ時間を待った後（ステップS13）、ステップS11に戻る。

【0036】一方、ホームページへのアクセスがあった場合は、OODB4内に未処理の小ジョブがあるかどうかを判断する（ステップS12）。ここで、小ジョブ#1～#nの配信が全て終わるなどして未処理の小ジョブ

が残っていない場合は、所定のディレイ時間を待った後（ステップ S13）、ステップ S11に戻る。また、未処理の小ジョブがまだ残っている場合は、ホームページへのアクセスを行った端末に対してその未処理の小ジョブを送信する（ステップ S14）。

【0037】このステップ S14では、まず、ホームページのアクセスを行った端末に対してサーバ1からエージェント（Applet）をダウンロードする。そして、そのエージェントが当該端末に常駐して、定められたプログラムに従ってサーバ1に対してジョブ取得要求を与えることにより、未処理の小ジョブをサーバ1から取得する。小ジョブの配信を受けた端末では、その取得した小ジョブの処理を実行し、その処理結果である回答をサーバ1に返すことになる。サーバ1内の回答管理部15は、受け取った回答を OODB 4内に格納するとともに、図2に示した回答管理テーブルを更新する。

【0038】以上のように、本実施形態では、1つの計算機で処理すると大きな負荷がかかる大規模ジョブ3を複数の小ジョブ #1～#n に分割し、それぞれを複数の端末 2a～2e に配信して分散実行させることにより、ネットワーク上で密結合された各端末 2a～2e の CPU 群を有効に利用して少ない負荷で大規模ジョブ3の処理を実行することができ、計算時間を短縮することもできる。

【0039】また、本実施形態では、各端末 2a～2e に対して小ジョブ #1～#n を強制的に配信して実行させるのではなく、端末 2a～2e からホームページへのアクセスがあったときにのみ、つまり、各端末 2a～2e のユーザからジョブ実行の意志が伝えられたときにのみ小ジョブ #1～#n を配信して実行させるようにしている。各端末 2a～2e の CPU パワーを、いつ実行されるかわからない大規模ジョブ3に対して常に開放しておかなくても済む。

【0040】例えば、端末 2a～2e 上での自己の業務等を休止している昼休みや夜間などにサーバ1のホームページにアクセスすることにより、各端末 2a～2e 上で通常の処理を行う際にそれらの CPU に余計な負担をかけないようにすることができる。なお、この場合、エージェントのプログラム中に、小ジョブの配信を受けて処理するという動作を昼休みや夜間などの時間帯で実行せよ、という内容を記述しておくことも可能である。このようにすれば、ホームページにいつアクセスしたかわからず、CPU パワーが余っている時間帯に常に分散処理を実行することができるようになる。

【0041】また、本実施形態では、大規模ジョブ3の分散処理をオブジェクト指向アーキテクチャに基づき実行するようにしており、特に、分割された小ジョブ #1～#n やそれらの回答をオブジェクト指向の OODB 4に格納するようにしている。これにより、リモートでの処理が行いやすくなるだけでなく、分散処理の途中で何

らかのトラブルによってシステムがダウンしたとしても、それまでに得られた回答は OODB 4内に確実に保管されるため、分散処理を再開するときに最初からやり直さなくても済み、処理の無駄を防止できる。

【0042】なお、上記実施形態では、ジョブの管理を行うサーバ1が、ユーザが通常使用する端末 2a～2e とは別に設けられているが、このサーバ1が持つ機能を、ネットワーク上に分散して存在する各端末 2a～2e のうちの1つが備えるようにしても良い。また、上記実施形態では、ホームページへのアクセスをトリガとして小ジョブ #1～#n の配信を実行しているが、このトリガは、必ずしもホームページへのアクセスである必要はなく、端末 2a～2e からサーバ1に対して要求が伝えられるものであれば何でも良い。

【0043】また、上記実施形態では詳しく説明していないが、大規模ジョブ3を複数の小ジョブ #1～#n に分割して分散処理し、それぞれの端末 2a～2e から得た回答をどのように処理するかについては、アプリケーションの内容に応じて適宜決めれば良い。各端末 2a～2e から得られた個々の回答をそのまま利用しても良いし、それらを1つの回答にまとめ上げるようにしても良い。どのようにまとめ上げるかについては、大規模ジョブ3のプログラム中に分割方法を記述したのと同様、大規模ジョブ3のプログラム中に記述される。

【0044】また、上記実施形態で述べた Applet 方式では、インターネット上の不特定のユーザからサーバ1にアクセスされる可能性があるという特徴を生かして、各端末 2a～2e に配信された小ジョブ #1～#n の実行をバックグラウンド処理にて行わせ、ブラウザを見ているユーザからは計算過程を意識させないようにする。なお、これとは逆に、配信されたジョブの実行中であることをユーザに知らせるために、例えば「ジョブ実行中」等のメッセージを画面上に表示するようにしても良い。

【0045】（第2の実施形態）次に、本発明の第2の実施形態について説明する。図5は、第2の実施形態による分散処理システムの構成を示すブロック図であり、図1に示したブロックと同じブロックには同一の符号を付している。第2の実施形態では、上述した第1の実施形態と比べて、サーバ1内に端末管理部17が更に設けられている。この端末管理部17は、図6に示すような端末管理テーブルを有する。以下、図6について説明する。

【0046】図6に示すように、端末管理部17は、サーバ1から端末 2a～2e に対して最初に基本となるジョブを送ってからその回答が返ってくるまでの応答時間をそれぞれの端末毎に管理し、その応答時間に応じて、次回から一度に与えるジョブ数を設定する。応答時間が短い場合は、その相手先端末の CPU 能力が高いと考えられるので、一度に送るジョブ数を多くする。一方、応答時間が長い場合は、その相手先端末の CPU 能力は低

いと考えられるので、一度に送るジョブ数を少なくする。また、各ジョブそのものの負荷が異なるのであれば、サーバ1で各ジョブ毎の負荷を管理し、CPU能力が高いと思われる端末には負荷の重いジョブを与えるようにしても良い。

【0047】このように、本実施形態では、ジョブを送ってからその回答が返ってくるまでの応答時間に応じて、処理能力の高いCPUを備えた端末に対して多くの分割ジョブを割り振るようなダイナミック管理を行うようにしたので、大規模ジョブ3の分散処理をより効率的に行うことができる。また、複数の小ジョブを同時に送ることになるので、全ての小ジョブを1つ1つ送っていた第1の実施形態に比べて、ネットワーク通信の処理負荷を減らすこともできる。

【0048】なお、図6中に示される端末IDは、ネットワーク上に存在する各端末2a~2eを特定するための識別番号であるが、本実施形態の場合、インターネットを介して不特定の端末からもアクセスされることがあるため、端末IDとしてはIPアドレスを用いるのが好ましい。なお、インターネットには接続せず、閉じたLANやWANのみを用いるのであれば、対象とする端末を全てサーバ1側で最初から把握できるので、必ずしもIPアドレスを用いる必要はない。

【0049】(第3の実施形態)次に、本発明の第3の実施形態について説明する。図7は、第3の実施形態による分散処理システムの構成を示すブロック図である。本実施形態においても、図7のサーバ1は、分割可能な大規模ジョブ3を適宜分割し、ネットワーク上に存在する複数の端末2a~2eに対して分割済の小さなジョブ#1~#nを割り当てて解いてもらう。そして、それぞれの端末2a~2eから小ジョブ#1~#nの処理結果である回答を集めて、最初の大規模ジョブ3の解答とする。ただし、このような動作の実現方法として、第3の実施形態では、上記第1の実施形態で説明したApplet方式とは異なり、JAVA言語のRMI方式を採用する。

【0050】図7に示すように、本実施形態のサーバ1には、図1に示す第1の実施形態では備えられていたアクセス受信部12がなく、その代わりにRMI端末管理部18が設けられている。また、各端末2a~2eには、RMI宣言部25が更に設けられている。図7中のその他の構成ブロックで、図1に示した符号と同一の符号を付したものは、同一の機能を有するものであるもので、これについての詳細な説明は省略する。

【0051】上記RMI宣言部25は、分割された小ジョブ#1~#nをRMI方式で受け取って実行するという宣言を行うためのものである。RMI宣言をしていると、その端末に対してサーバ1から分割された小ジョブ#1~#nが送られてきて、当該端末において大規模ジョブ3の一部が実行されることになる。RMI宣言は、例えば、電源投入時の初期プログラムによって行われ

る。また、その後必要に応じて、ユーザがRMI宣言を取り止めたり、再び宣言をしたりすることも自由にできるものである。

【0052】一方、サーバ1内のRMI端末管理部18は、ネットワークを介してサーバ1に接続されている各端末2a~2eのうち、RMI宣言をしている端末がどれであるかを管理する。どの端末がRMI宣言をしているかを把握するための方法としては、様々な方法が考えられる。例えば、サーバ1から各端末2a~2eに対して定期的に確認信号を送り、その応答として各端末2a~2eが、現在RMI宣言をしているかどうかを返信するようにする方法が考えられる。このとき、所定のタイムアウト時間を過ぎてもある端末から応答がない場合は、その端末は電源が切られているなどによって現在使用できない状態であると判断する。

【0053】また、ユーザが端末2a~2e上でRMI宣言の設定状態を変更する毎に、そのことを端末2a~2e側からサーバ1に能動的に通知するようにしても良い。このようにすれば、各端末2a~2eにおける実際のRMI宣言の設定状態と、サーバ1側で管理している各端末2a~2eのRMI宣言の状態とをリアルタイムで一致させることができる。

【0054】この実施形態でエージェントは、RMI宣言をした時点でその端末上に立ち上がり、常駐するものである。このエージェントの機能は、第1の実施形態で述べたものと同様であり、少なくとも小ジョブの取得機能と回答の返信機能とを有する。

【0055】次に、以上のように構成した第3の実施形態による分散処理システム中のサーバ1の動作を、図8のフローチャートを参照しながら説明する。なお、サーバ1の全体の動作は、図3のフローチャートに示したものと同様であり、図8のフローチャートは、図3中のステップS5における動作の中身を示している。なお、本実施形態の場合、図3のステップS1では、図2に示したような回答管理テーブルを初期状態にセットする処理の他に、どの端末がRMI方式で使えるかを表したテーブルをセットする処理等も行う。

【0056】図8において、まず最初に、サーバ1は、ODDB4内に未処理の小ジョブがあるかどうかを判断する(ステップS21)。ここで、小ジョブ#1~#nの配信が全て終わるなどして未処理の小ジョブが残っていない場合は、所定のディレイ時間を待った後(ステップS23)、ステップS21に戻る。また、未処理の小ジョブがまだ残っている場合は、利用できるRMI処理端末があるかどうかを更に判断する(ステップS22)。

【0057】ここで、利用できるRMI処理端末とは、例えば、RMI宣言がされていて、かつ、現在小ジョブの処理を実行していない端末のことを言う。利用できるRMI処理端末がない場合は、所定のディレイ時間を待

った後(ステップS23)、ステップS21に戻る。一方、利用できるRMI処理端末がある場合は、OODB4に格納されている複数の小ジョブ#1~#nのうち未処理の小ジョブを、当該利用できるRMI処理端末の1つに対して送信する(ステップS24)。

【0058】このステップS24では、利用できるRMI処理端末に常駐しているエージェントが、定められたプログラムに従ってサーバ1に対してジョブ取得要求を与えることにより、未処理の小ジョブをサーバ1から取得する。小ジョブの配信を受けた端末では、その取得した小ジョブの処理を実行し、その処理結果である回答をサーバ1に返すことになる。

【0059】このように、第3の実施形態では、端末2a~2eからアクセスがあるのを待ってジョブ配信を行う第1の実施形態とは異なり、小ジョブ#1~#nを振り分ける側のサーバ1から積極的に、現在CPUが空いていて利用可能な端末があるかどうかをチェックしてジョブ配信を行っている。その他の点は、第1の実施形態と同様である。したがって、第3の実施形態においても、少ない計算負荷で大規模ジョブ3の処理を実行することができる。

【0060】また、本実施形態では、各端末2a~2eに対して小ジョブ#1~#nを強制的に配信して実行させるのではなく、端末2a~2eでRMI宣言をしているときにのみ小ジョブ#1~#nを配信して実行させるようにしているので、各端末2a~2eのCPUパワーを、いつ実行されるか分からない大規模ジョブ3に対して常に開放しておかなくても済む。また、本実施形態でも、分散処理システムをオブジェクト指向に基づき構築しているので、リモートでの処理が行いやすくなるだけでなく、何らかのトラブルでシステムがダウンしたときの処理の無駄を防止することができる。

【0061】なお、上記第3の実施形態では、サーバ1内にRMI端末管理部18を設け、RMI宣言をしている端末がどれであるかを管理するようにしているが、ジョブ配信に先立って、まず最初にサーバ1から端末2a~2eに対して問い合わせ信号等を送信し、その応答を見ることによってRMI宣言されているかどうかを確認するようにすれば、サーバ1は、RMI宣言をしている端末がどれであるかの管理テーブルを持つ必要がなくなる。

【0062】また、上記第3の実施形態では、図8のステップS22で説明したように、RMI宣言をしておき、かつ、現在小ジョブの処理を実行していない端末に対してのみ小ジョブを割り振るようにしたが、小ジョブの処理を実行中の端末に対しても新たな小ジョブを割り当てるようにしても良い。例えば、第2の実施形態で述べたように、送信先の端末のCPU能力によっては、複数のジョブを割り当てても十分なスピードで処理できる場合もあるので、そのような場合には現在実行中の小

ジョブがあっても更に次の小ジョブを割り当てるようにする。これは、図7に示した構成に対して、図5に示した端末管理部17を更に設けることによって実現することができる。

【0063】また、上記第3の実施形態では、各端末2a~2eのCPUを利用する方法として、JAVA言語のRMI方式を採用したが、CORBA(Common Object Request Broker Architecture)方式も規格化されており、これをフレームワークに組み入れるようにしても良い。また、この第3の実施形態でも、ジョブの管理を行うサーバ1が、ユーザが通常使用する端末2a~2eとは別に設けられているが、このサーバ1が持つ機能を、ネットワーク上に分散して存在する各端末2a~2eのうちの1つが備えるようにしても良い。

【0064】(第4の実施形態)次に、本発明の第4の実施形態について説明する。上記第1~第3の実施形態では、Applet方式あるいはRMI方式をそれぞれ単独で用いてきたが、これらの方式を併用するようにしても良い。この場合、分散処理システムの構成は、例えば、図1、図5および図7に示した機能ブロックを全て備えたものとすれば良い。また、図4に示しフローチャートと、図8に示したフローチャートは、サーバ1側で常に並列的に動作していることになる。

【0065】この場合、例えば、RMI方式で取得した小ジョブの処理中に、Applet方式で新たな小ジョブを取得して実行することも可能とするなど、高い自由度を持たせることも可能であるが、大規模ジョブ3の分散処理をより効率化させるために何らかの制約を設けるようにしても良い。例えば、その端末が備えるCPUの能力に応じて、一方の方式で小ジョブを実行している間は、他の方式で小ジョブを更に実行することを禁止するなどの制約を設けることが可能である。

【0066】図9および図10は、Applet方式とRMI方式の併存時に制約を加えるために必要なテーブル情報の例を示すものであり、例えば端末管理部17が備える。図9に示す端末管理テーブルは、図6に示したものとほぼ同様であるが、現在その端末がどの方式で動作しているか、つまり、小ジョブの配信を端末2a~2eからの要求に応じて受動的に行うApplet方式、小ジョブの配信を利用可能な状態にある端末2a~2eに対して能動的に行うRMI方式、あるいはその両方式の何れで動作しているかを管理するための項目が設けられている。

【0067】また、図10に示す端末設定テーブルは、それぞれの端末2a~2eが、上記Applet方式、RMI方式、あるいはその両方式のどの方式で動作し得るかをあらかじめ設定したものである。例えば、図10中の一番上に示された端末は、Applet方式でしか動作できないように設定されている。また、上から3番目に示された端末は、Applet方式でもRMI方式でも動作できるが、両方の方式を同時には実行できないように設定されてい

る。サーバ 1 は、上記図 9 中の現状を表す情報と、図 10 のように設定された内容とを参照して、端末 2 a ~ 2 e に対する小ジョブの割り振りに制約を与えることになる。なお、大規模ジョブ 3 を分割する際に、Applet 方式、RMI 方式の何れ的方式にも対応するように小ジョブ # 1 ~ # n を作成することが好ましい。

【0068】（本発明の他の実施形態）なお、以上に説明した本実施形態の分散処理システムが備える各機能ブロックは、実際にはコンピュータの CPU あるいは MPU、RAM、ROM などで構成されるものであり、RAM や ROM に記憶されたプログラムが動作することによって実現できる。したがって、コンピュータが上記機能を果たすように動作させるプログラムを、例えば CD-ROM のような記録媒体に記録し、コンピュータに読み込ませることによって実現できるものである。記録媒体としては、CD-ROM 以外に、フロッピーディスク、ハードディスク、磁気テープ、光磁気ディスク、不揮発性メモリカード等を用いることができる。

【0069】また、コンピュータが供給されたプログラムを実行することにより上述の実施形態の機能が実現されるだけでなく、そのプログラムがコンピュータにおいて稼働している OS（オペレーティングシステム）あるいは他のアプリケーションソフト等と共同して上述の実施形態の機能が実現される場合や、供給されたプログラムの処理の全てあるいは一部がコンピュータの機能拡張ボードや機能拡張ユニットにより行われて上述の実施形態の機能が実現される場合も、かかるプログラムは本発明の実施形態に含まれる。

【0070】なお、上記に示した各実施形態は、何れも本発明を実施するにあたっての具体化の一例を示したものに過ぎず、これらによって本発明の技術的範囲が限定的に解釈されてはならないものである。すなわち、本発明はその精神、またはその主要な特徴から逸脱することなく、様々な形で実施することができる。

【0071】

【発明の効果】本発明は上述したように、ジョブ取得要求が行われた計算機または利用可能な状態にある計算機に対し、分割された複数のジョブの中から未処理のジョブを与え、与えられた分割ジョブを各計算機上で処理してその回答を当該分割ジョブの供給元に返すように構成したので、1つの計算機で処理すると大きな負荷がかかる大規模な処理を、複数の計算機を利用して少ない処理負荷で実行することができる。また、本発明では、各計算機に対して分割ジョブを強制的に配信して実行させるのではなく、計算機からジョブ取得要求があったとき、あるいは利用可能な状態にある計算機に対してのみ、分割ジョブを配信して実行させるようにしているので、各計算機の CPU パワーを、いつ実行されるか分からない大規模な処理に対して常に開放しておかなくても済み、各計算機に余計な負担をかけないようにすることができ

る。

・【0072】本発明の他の特徴によれば、計算機の状態に応じて未処理ジョブの配信を制御するようにした。例えば、計算機に分割ジョブを送信してからその回答が返ってくるまでの応答時間に応じて、与えるジョブ数を可変とするようにしたり、分割ジョブの配信を行う際の各種方式の計算機上での採用状態に応じて、分割ジョブの配信に制約を与えるようにしたので、大規模ジョブの分散処理をより効率的に行うことができる。

10 【図面の簡単な説明】

【図 1】第 1 の実施形態による分散処理システムの構成を示すブロック図である。

【図 2】本実施形態の回答管理部が備える回答管理テーブルの例を示す図である。

【図 3】本実施形態による分散処理システム中のサーバが行う全体の動作の流れを示すフローチャートである。

【図 4】Applet 方式の場合に、図 3 中のステップ S 5 において行われる動作の中身を示すフローチャートである。

20 【図 5】第 2 の実施形態による分散処理システムの構成を示すブロック図である。

【図 6】第 2 の実施形態による端末管理部が備える端末管理テーブルの例を示す図である。

【図 7】第 3 の実施形態による分散処理システムの構成を示すブロック図である。

【図 8】RMI 方式の場合に、図 3 中のステップ S 5 において行われる動作の中身を示すフローチャートである。

30 【図 9】Applet 方式と RMI 方式の併存時に制約を加えるために必要なテーブル情報の例を示す図である。

【図 10】Applet 方式と RMI 方式の併存時に制約を加えるために必要なテーブル情報の例を示す図である。

【符号の説明】

1 サーバ（ジョブ管理端末装置）

2 a ~ 2 e 端末

3 大規模ジョブ

4 OODB

11 ジョブ分割部

12 アクセス受信部

40 13 エージェント送信部

14 ジョブ送信部

15 回答管理部

16 CPU

17 端末管理部

18 RMI 端末管理部

21 要求／回答送信部

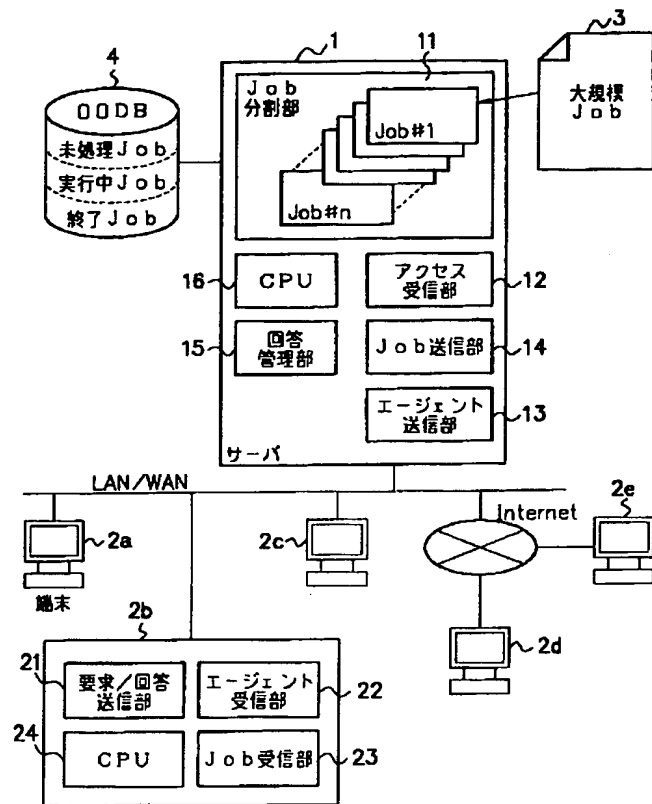
22 エージェント受信部

23 ジョブ受信部

24 CPU

50 25 RMI 宣言部

【図 1】



【図 2】

回答管理テーブル タイムアウト=5min

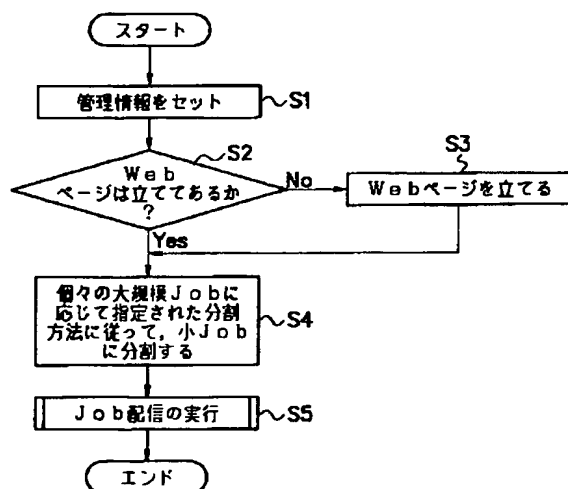
Job ID	Job送信時刻	回答受信時刻	終了
No.1	11:00	11:02	O
No.2	11:01		X
No.3	11:02	11:04	O
No.4	11:03		
⋮	⋮	⋮	⋮

【図 6】

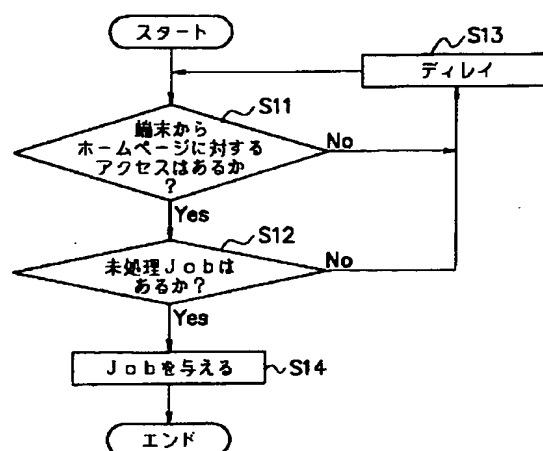
端末管理テーブル

端末ID (IPアドレス)	応答時間	可能配信Job数
10,100,100,100	3min	2
10,100,100,101	1min	5
10,100,100,102	5min	1
⋮	⋮	⋮

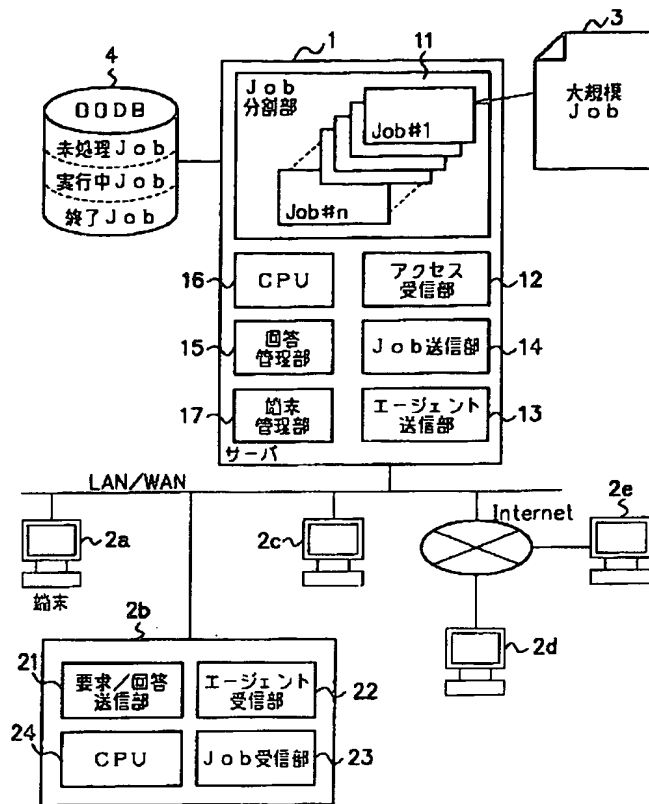
【図 3】



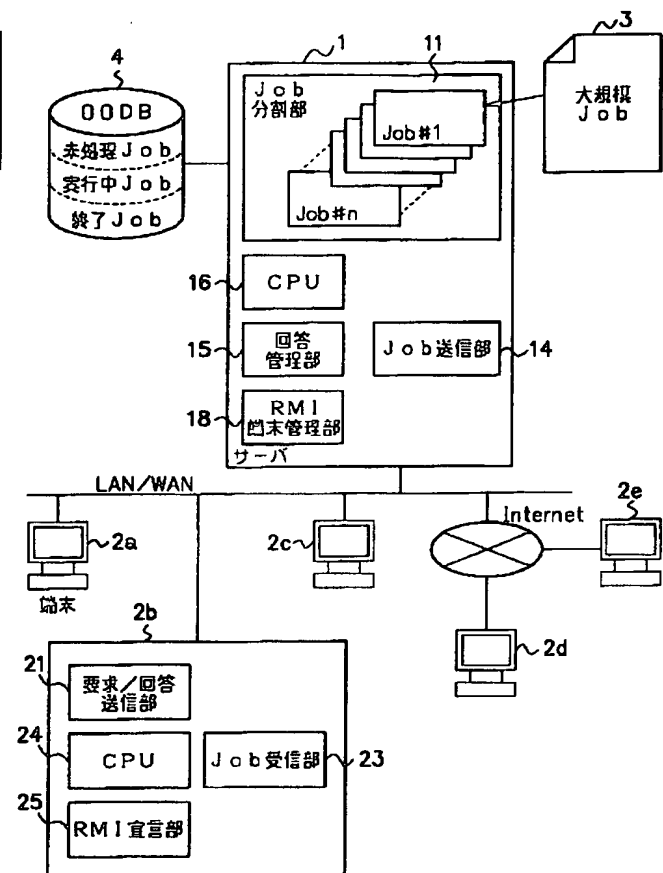
【図 4】



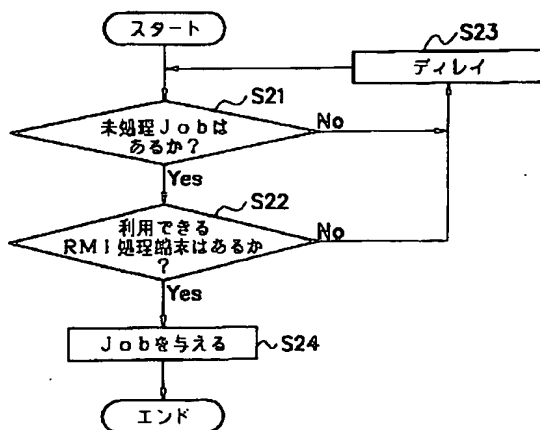
【図 5】



【図 7】



【図 8】



【図 9】

端末管理テーブル

IP 本 ID (IP アドレス)	応答時間	可能配信 Job 数	現状
10,100,100,100	3min	2	Applet
10,100,100,101	1min	5	Applet/RMI
10,100,100,102	5min	1	RMI
...

・【図 10】

端末設定テーブル

端末ID (IPアドレス)	Applet	RMI	Applet/RMI
10,100,100,100	○	×	×
10,100,100,101	○	○	○
10,100,100,102	○	○	×
10,100,100,103	×	○	×
⋮	⋮	⋮	⋮

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.